



PPPPPPPPP LL IIIIIII HH HH EEEEEEEEEE EEEEEEEEEE PPPPPPPP  
PPPPPPPPP LL IIIIIII HH HH EEEEEEEEEE EEEEEEEEEE PPPPPPPP  
PP PP LL II HH HH EE EE PP PP  
PP PP LL II HH HH EE EE PP PP  
PP PP LL II HH HH EE EE PP PP  
PP PP LL II HH HH EE EE PP PP  
PPPPPPPPP LL II HHHHHHHHHHH EEEEEEEE EEEEEEEE PPPPPPPP  
PPPPPPPPP LL II HHHHHHHHHHH EEEEEEEE EEEEEEEE PPPPPPPP  
PP LL II HH HH EE EE PP  
PP LLLLLLLL IIIIIII HH HH EEEEEEEEEE EEEEEEEEEE PP  
PP LLLLLLLL IIIIIII HH HH EEEEEEEEEE EEEEEEEEEE PP

The diagram illustrates a 10x10 grid of binary symbols, likely representing a feature map from a convolutional neural network. The symbols are organized into four distinct vertical columns. The first column contains ten 'L' symbols, each aligned vertically. The second column contains ten 'I' symbols, also aligned vertically. The third column contains ten 'S' symbols, aligned vertically. The fourth column contains ten 'S' symbols, aligned vertically. This visual representation highlights the spatial structure and feature extraction process in a neural network.

(1) 64 pli\$alloheap - allocate heap space  
(1) 118 pli\$freeheep - deallocate heap space  
(1) 157 pli\$allocation - return number of time that has been allocated

```
0000 1 .title plisheep - pl1 runtime heap allocation
0000 2 .ident /1-003/
0000 3 ****
0000 4 ****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 ****
0000 26 *
0000 27 ++
0000 28 facility:
0000 29
0000 30 VAX/VMS PL1 runtime library.
0000 31
0000 32 abstract:
0000 33
0000 34 Runtime routines to allocate and deallocate heap space.
0000 35
0000 36 author: R. Heinen 16-dec-1978
0000 37
0000 38 modifications:
0000 39
0000 40 1-002
0000 41 Modified both routines, to allow them be called with different
0000 42 number of parameters and so work for the controlled variable.
0000 43 Alex Wu 03/24/82
0000 44
0000 45 add the new routine - PLI$ALLOCATION to return the number of
0000 46 generation that a controlled variable has been allocated.
0000 47 Alex Wu 03/30/82
0000 48
0000 49 1-003 Bill Matthews 29-September-1982
0000 50
0000 51 Invoke macros $defdat and rtshare instead of $defopr and share.
0000 52
0000 53
0000 54 external definitions
0000 55
0000 56
0000 57
```

PLI\$HEEP  
1-003

- pl1 runtime heap allocation

H 6

16-SEP-1984 02:21:23 VAX/VMS Macro V04-00  
6-SEP-1984 11:38:35 [PLIRTL.SRC]PLIHEEP.MAR;1

Page 2  
(1)

0000 58 :  
0000 59 : local data  
0000 60 :  
0000 61 :  
0000 62 rtshare

```

0000 64 .sbttl pli$aloheap - allocate heap space
0000 65 :++
0000 66 : pli$aloheap - allocate heap space
0000 67 :
0000 68 : functional description:
0000 69 :
0000 70 : This routine allocates a memory block using "lib$get_vm".
0000 71 : The allocated block is a longword (or two longword)-bigger in order to
0000 72 : save the allocated size (and/or the previous pointer) in the block itself.
0000 73 : The size is stored in the first longword (the pointer is save in the first
0000 74 : longword). The returned address
0000 75 : is the address of the actual free space.
0000 76 :
0000 77 : inputs:
0000 78 :
0000 79 :     (ap) = 2 (or 3 for the controlled variables)
0000 80 :     4(ap) = longword size
0000 81 :     8(ap) = address to store address of the data
0000 82 :     12(ap) = address to the current block -- if exist
0000 83 :
0000 84 : outputs:
0000 85 :
0000 86 :     r0 = success indicator fixed binary(32)
0000 87 :
0000 88 :     r0 = ss$normal for success
0000 89 :         lib$insvirmem - insufficient memory for request
0000 90 :         lib$badblosiz - bad size parameter
0000 91 :
0000 92 :     If an error is indicated by r0 then the returned address is 0.
0000 93 :     and the error is signalled.
0000 94 :-- .entry pli$aloheap,0
0000 95 :
0000 96 :
0000 97 :     clrl    a8(ap) : assume allocation failure
0000 98 :     addl3   #4,4(ap),-(sp) : build following arg list
0000 99 :     cmpl    #2,(ap) : based ?
0000 100:     beql    10$ : less than or equal then yes
0000 101:     addl2   #4,(sp) : one more longword for pointer
0000 102:     clrl    -(sp) : address to return address
0000 103:     pushab   (sp) :
0000 104:     pushab   8(sp) :
0000 105:     calls   #2,g^lib$get_vm : allocate the memory
0000 106:     blbc    r0,20$ : if low clear then error
0000 107:     popl    r0 : get address of allocated space
0000 108:     popl    (r0)+ : insert size in buffer
0000 109:     cmpl    #2,(ap) : based ?
0000 110:     beql    15$ : less than or equal than yes
0000 111:     movl    12(ap),(r0)+ : save pointer
0000 112:     movl    r0,a8(ap) : return address of allocated memory
0000 113:     ret :
0000 114:     pushl    r0 : signal error condition
0000 115:     calls   #1,g^lib$signal :
0000 116:     ret :
0000 117:     .end
0000 118:     .end
0000 119:     .end
0000 120:     .end
0000 121:     .end
0000 122:     .end
0000 123:     .end
0000 124:     .end
0000 125:     .end
0000 126:     .end
0000 127:     .end
0000 128:     .end
0000 129:     .end
0000 130:     .end
0000 131:     .end
0000 132:     .end
0000 133:     .end
0000 134:     .end
0000 135:     .end
0000 136:     .end
0000 137:     .end
0000 138:     .end
0000 139:     .end
0000 140:     .end
0000 141:     .end
0000 142:     .end
0000 143:     .end
0000 144:     .end
0000 145:     .end
0000 146:     .end
0000 147:     .end
0000 148:     .end
0000 149:     .end
0000 150:     .end
0000 151:     .end
0000 152:     .end
0000 153:     .end
0000 154:     .end
0000 155:     .end
0000 156:     .end
0000 157:     .end
0000 158:     .end
0000 159:     .end
0000 160:     .end
0000 161:     .end
0000 162:     .end
0000 163:     .end
0000 164:     .end
0000 165:     .end
0000 166:     .end
0000 167:     .end
0000 168:     .end
0000 169:     .end
0000 170:     .end
0000 171:     .end
0000 172:     .end
0000 173:     .end
0000 174:     .end
0000 175:     .end
0000 176:     .end
0000 177:     .end
0000 178:     .end
0000 179:     .end
0000 180:     .end
0000 181:     .end
0000 182:     .end
0000 183:     .end
0000 184:     .end
0000 185:     .end
0000 186:     .end
0000 187:     .end
0000 188:     .end
0000 189:     .end
0000 190:     .end
0000 191:     .end
0000 192:     .end
0000 193:     .end
0000 194:     .end
0000 195:     .end
0000 196:     .end
0000 197:     .end
0000 198:     .end
0000 199:     .end
0000 200:     .end
0000 201:     .end
0000 202:     .end
0000 203:     .end
0000 204:     .end
0000 205:     .end
0000 206:     .end
0000 207:     .end
0000 208:     .end
0000 209:     .end
0000 210:     .end
0000 211:     .end
0000 212:     .end
0000 213:     .end
0000 214:     .end
0000 215:     .end
0000 216:     .end
0000 217:     .end
0000 218:     .end
0000 219:     .end
0000 220:     .end
0000 221:     .end
0000 222:     .end
0000 223:     .end
0000 224:     .end
0000 225:     .end
0000 226:     .end
0000 227:     .end
0000 228:     .end
0000 229:     .end
0000 230:     .end
0000 231:     .end
0000 232:     .end
0000 233:     .end
0000 234:     .end
0000 235:     .end
0000 236:     .end
0000 237:     .end
0000 238:     .end
0000 239:     .end
0000 240:     .end
0000 241:     .end
0000 242:     .end
0000 243:     .end
0000 244:     .end
0000 245:     .end
0000 246:     .end
0000 247:     .end
0000 248:     .end
0000 249:     .end
0000 250:     .end
0000 251:     .end
0000 252:     .end
0000 253:     .end
0000 254:     .end
0000 255:     .end
0000 256:     .end
0000 257:     .end
0000 258:     .end
0000 259:     .end
0000 260:     .end
0000 261:     .end
0000 262:     .end
0000 263:     .end
0000 264:     .end
0000 265:     .end
0000 266:     .end
0000 267:     .end
0000 268:     .end
0000 269:     .end
0000 270:     .end
0000 271:     .end
0000 272:     .end
0000 273:     .end
0000 274:     .end
0000 275:     .end
0000 276:     .end
0000 277:     .end
0000 278:     .end
0000 279:     .end
0000 280:     .end
0000 281:     .end
0000 282:     .end
0000 283:     .end
0000 284:     .end
0000 285:     .end
0000 286:     .end
0000 287:     .end
0000 288:     .end
0000 289:     .end
0000 290:     .end
0000 291:     .end
0000 292:     .end
0000 293:     .end
0000 294:     .end
0000 295:     .end
0000 296:     .end
0000 297:     .end
0000 298:     .end
0000 299:     .end
0000 300:     .end
0000 301:     .end
0000 302:     .end
0000 303:     .end
0000 304:     .end
0000 305:     .end
0000 306:     .end
0000 307:     .end
0000 308:     .end
0000 309:     .end
0000 310:     .end
0000 311:     .end
0000 312:     .end
0000 313:     .end
0000 314:     .end
0000 315:     .end
0000 316:     .end
0000 317:     .end
0000 318:     .end
0000 319:     .end
0000 320:     .end
0000 321:     .end
0000 322:     .end
0000 323:     .end
0000 324:     .end
0000 325:     .end
0000 326:     .end
0000 327:     .end
0000 328:     .end
0000 329:     .end
0000 330:     .end
0000 331:     .end
0000 332:     .end
0000 333:     .end
0000 334:     .end
0000 335:     .end
0000 336:     .end
0000 337:     .end
0000 338:     .end
0000 339:     .end
0000 340:     .end
0000 341:     .end
0000 342:     .end
0000 343:     .end
0000 344:     .end
0000 345:     .end
0000 346:     .end
0000 347:     .end
0000 348:     .end
0000 349:     .end
0000 350:     .end
0000 351:     .end
0000 352:     .end
0000 353:     .end
0000 354:     .end
0000 355:     .end
0000 356:     .end
0000 357:     .end
0000 358:     .end
0000 359:     .end
0000 360:     .end
0000 361:     .end
0000 362:     .end
0000 363:     .end
0000 364:     .end
0000 365:     .end
0000 366:     .end
0000 367:     .end
0000 368:     .end
0000 369:     .end
0000 370:     .end
0000 371:     .end
0000 372:     .end
0000 373:     .end
0000 374:     .end
0000 375:     .end
0000 376:     .end
0000 377:     .end
0000 378:     .end
0000 379:     .end
0000 380:     .end
0000 381:     .end
0000 382:     .end
0000 383:     .end
0000 384:     .end
0000 385:     .end
0000 386:     .end
0000 387:     .end
0000 388:     .end
0000 389:     .end
0000 390:     .end
0000 391:     .end
0000 392:     .end
0000 393:     .end
0000 394:     .end
0000 395:     .end
0000 396:     .end
0000 397:     .end
0000 398:     .end
0000 399:     .end
0000 400:     .end
0000 401:     .end
0000 402:     .end
0000 403:     .end
0000 404:     .end
0000 405:     .end
0000 406:     .end
0000 407:     .end
0000 408:     .end
0000 409:     .end
0000 410:     .end
0000 411:     .end
0000 412:     .end
0000 413:     .end
0000 414:     .end
0000 415:     .end
0000 416:     .end
0000 417:     .end
0000 418:     .end
0000 419:     .end
0000 420:     .end
0000 421:     .end
0000 422:     .end
0000 423:     .end
0000 424:     .end
0000 425:     .end
0000 426:     .end
0000 427:     .end
0000 428:     .end
0000 429:     .end
0000 430:     .end
0000 431:     .end
0000 432:     .end
0000 433:     .end
0000 434:     .end
0000 435:     .end
0000 436:     .end
0000 437:     .end
0000 438:     .end
0000 439:     .end
0000 440:     .end
0000 441:     .end
0000 442:     .end
0000 443:     .end
0000 444:     .end
0000 445:     .end
0000 446:     .end
0000 447:     .end
0000 448:     .end
0000 449:     .end
0000 450:     .end
0000 451:     .end
0000 452:     .end
0000 453:     .end
0000 454:     .end
0000 455:     .end
0000 456:     .end
0000 457:     .end
0000 458:     .end
0000 459:     .end
0000 460:     .end
0000 461:     .end
0000 462:     .end
0000 463:     .end
0000 464:     .end
0000 465:     .end
0000 466:     .end
0000 467:     .end
0000 468:     .end
0000 469:     .end
0000 470:     .end
0000 471:     .end
0000 472:     .end
0000 473:     .end
0000 474:     .end
0000 475:     .end
0000 476:     .end
0000 477:     .end
0000 478:     .end
0000 479:     .end
0000 480:     .end
0000 481:     .end
0000 482:     .end
0000 483:     .end
0000 484:     .end
0000 485:     .end
0000 486:     .end
0000 487:     .end
0000 488:     .end
0000 489:     .end
0000 490:     .end
0000 491:     .end
0000 492:     .end
0000 493:     .end
0000 494:     .end
0000 495:     .end
0000 496:     .end
0000 497:     .end
0000 498:     .end
0000 499:     .end
0000 500:     .end
0000 501:     .end
0000 502:     .end
0000 503:     .end
0000 504:     .end
0000 505:     .end
0000 506:     .end
0000 507:     .end
0000 508:     .end
0000 509:     .end
0000 510:     .end
0000 511:     .end
0000 512:     .end
0000 513:     .end
0000 514:     .end
0000 515:     .end
0000 516:     .end
0000 517:     .end
0000 518:     .end
0000 519:     .end
0000 520:     .end
0000 521:     .end
0000 522:     .end
0000 523:     .end
0000 524:     .end
0000 525:     .end
0000 526:     .end
0000 527:     .end
0000 528:     .end
0000 529:     .end
0000 530:     .end
0000 531:     .end
0000 532:     .end
0000 533:     .end
0000 534:     .end
0000 535:     .end
0000 536:     .end
0000 537:     .end
0000 538:     .end
0000 539:     .end
0000 540:     .end
0000 541:     .end
0000 542:     .end
0000 543:     .end
0000 544:     .end
0000 545:     .end
0000 546:     .end
0000 547:     .end
0000 548:     .end
0000 549:     .end
0000 550:     .end
0000 551:     .end
0000 552:     .end
0000 553:     .end
0000 554:     .end
0000 555:     .end
0000 556:     .end
0000 557:     .end
0000 558:     .end
0000 559:     .end
0000 560:     .end
0000 561:     .end
0000 562:     .end
0000 563:     .end
0000 564:     .end
0000 565:     .end
0000 566:     .end
0000 567:     .end
0000 568:     .end
0000 569:     .end
0000 570:     .end
0000 571:     .end
0000 572:     .end
0000 573:     .end
0000 574:     .end
0000 575:     .end
0000 576:     .end
0000 577:     .end
0000 578:     .end
0000 579:     .end
0000 580:     .end
0000 581:     .end
0000 582:     .end
0000 583:     .end
0000 584:     .end
0000 585:     .end
0000 586:     .end
0000 587:     .end
0000 588:     .end
0000 589:     .end
0000 590:     .end
0000 591:     .end
0000 592:     .end
0000 593:     .end
0000 594:     .end
0000 595:     .end
0000 596:     .end
0000 597:     .end
0000 598:     .end
0000 599:     .end
0000 600:     .end
0000 601:     .end
0000 602:     .end
0000 603:     .end
0000 604:     .end
0000 605:     .end
0000 606:     .end
0000 607:     .end
0000 608:     .end
0000 609:     .end
0000 610:     .end
0000 611:     .end
0000 612:     .end
0000 613:     .end
0000 614:     .end
0000 615:     .end
0000 616:     .end
0000 617:     .end
0000 618
```

```

0041 118 .sbttl pli$freeheap - deallocate heap space
0041 119 :++ pli$freeheap - deallocate heap space
0041 120 : functional description:
0041 121 :
0041 122 : This routine is the complementary routine to "pli$alocheep".
0041 123 : The memory is returned via "lib$free_vm".
0041 124 :
0041 125 :
0041 126 :
0041 127 : inputs:
0041 128 :
0041 129 :     (ap) = 1 (or 2 for the controlled variables)
0041 130 :     4(ap) = address of memory
0041 131 :     8(ap) = address of the based pointer -- if exist
0041 132 :
0041 133 : The block must have been allocated using "pli$alocheep".
0041 134 :
0041 135 : outputs:
0041 136 :
0041 137 :     For the controlled variables, there is a side effect on the
0041 138 :     based pointer @8(ap)
0041 139 :     r0 = success indicator fixed binary(32)
0041 140 :
0041 141 :     r0 = ss$ normal for success
0041 142 :     lib$badblosiz - incorrect block size parameter
0041 143 :-- .entry pli$freeheap,0
0000 0041 144 :
0000 0043 145 :
50 04 AC 04 C3 0043 146 subl3 #4,4(ap),r0      ; address block size or pointer longword
6C 01 D1 0048 147 cmpl #1,(ap)                 ; based variable ?
08 BC 07 13 004B 148 beql 30$                   ; less than or equal than yes
50 60 DD 004D 149 movl (r0),@8(ap)            ; save previous block pointer
50 04 C2 0051 150 subl2 #4,r0                  ; adress to block size
50 DD 0054 151 30$: pushl r0                    ; put address in memory
6E 9F 0056 152 pushab (sp)                  ; set up arg list
50 DD 0058 153 pushl r0                    ;
00000000'GF 02 FB 005A 154 calls #2,g^lib$free_vm ; free the memory
04 0061 155 ret

```

0062 157 .sbttl pli\$allocation - return number of time that has been allocated  
0062 158 :++  
0062 159 : pli\$allocation - return number of time that has been allocated  
0062 160 :  
0062 161 : functional description:  
0062 162 :  
0062 163 : This routine perform the builtin funtion ALLOCATION which return the  
0062 164 : generations that a controlled variable has been allocated. It walks  
0062 165 : through the link list and increment the count each time looks at a  
0062 166 : new block until the end of the list (null pointer).  
0062 167 :  
0062 168 : inputs:  
0062 169 :  
0062 170 : (ap) = 2  
0062 171 : 4(ap) = address of the current active block  
0062 172 : 8(ap) = address of the return value  
0062 173 :  
0062 174 : The block must have been allocated using "pli\$aloheep".  
0062 175 :  
0062 176 : outputs:  
0062 177 :  
0062 178 : 8(ap) contains the return value  
0062 179 :--  
0000 0062 180 .entry pli\$allocation,0  
0064 181  
0064 182  
0064 183  
0064 184  
0064 185 10\$:  
0064 186  
0064 187  
0064 188 30\$:  
0064 189  
0064 190 .end

50 08 BC D4 0064  
50 04 AC D0 0067  
08 13 006B  
50 08 BC D6 006D  
50 70 D0 0070  
F8 12 0073  
04 0075  
0076  
0076

182  
183  
184  
185  
186  
187  
188  
189  
190

clrl 08(ap) : initialize the counter  
movl 4(ap),r0 : get the pointer  
beql 30\$ : if null pointer then done  
incl 08(ap) : increment the counter  
movl -(r0),r0 : get previous block address  
bnequ 10\$ : keep looping if not null pointer  
ret

## PLISHEEP Symbol table

## - pl1 runtime heap allocation

L 6

16-SEP-1984 2:21:23 VAX/VMS Macro V04-00  
6-SEP-1984 11:38:55 [PLIRTL.SRC]PLIHEEP.MAR;1

Page 6  
(1)

LIBSFREE VM	*****	X	01
LIBSGET VM	*****	X	01
LIBSSIGNAL	*****	X	01
PLISALLOCATION	00000062	RG	01
PLISALOCHEEP	00000000	RG	01
PLISFREEHEEP	00000041	RG	01

+-----+  
! Psect synopsis !  
+-----+

**PSECT name**

### Allocation PSECT No. Attributes

• ABS 00000000 ( 0.) 00 ( 0.) NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE  
 PL\\$CODE 00000076 ( 118.) 01 ( 1.) PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.05	00:00:00.59
Command processing	68	00:00:00.54	00:00:01.51
Pass 1	61	00:00:00.55	00:00:01.26
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	43	00:00:00.38	00:00:00.60
Symbol table output	1	00:00:00.01	00:00:00.01
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	183	00:00:01.56	00:00:04.00

The working set limit was 750 pages.

2560 bytes (5 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 6 non-local and 6 local symbols.

190 source lines were read in Pass 1, producing 17 object records in Pass 2.

1 page of virtual memory was used to define 1 macro.

## Macro library statistics

### **Macro Library name**

## Macros defined

\$255\$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1  
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2  
TOTALS (all libraries)

3 GETS were required to define 1 macros.

There were no errors, warnings or information messages.

**MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS\$:PLIHEEP/OBJ=OBJ\$:PLIHEEP MSRC\$:PLIHEEP/UPDATE=(ENH\$:PLIHEEP)+LIB\$:PLIRTMAC/LIB**

0308 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

PLIFORMAT  
LTS

PLIGETBUF  
LTS

PLIGETEDI  
LTS

PLIHEEP  
LTS

PLIMSGTXT  
LTS

PLIPUTEFL  
LTS

PLIRMSBIS  
LTS

PLIRECPT  
LTS

PLIREAD  
LTS

PLIREWRT  
LTS

PLIPROTEC  
LTS

PLIPUTEDI  
LTS

PLIGETLIS  
LTS

PLIPUTLIS  
LTS

PLIPKD1UL  
LTS

PLIPKD1US  
LTS PLIPPUTBUF  
LTS

PLIGEFIL  
LTS

PLIMSGPTR  
LTS